**CS 504 – Software Engineering**

**HOP02 – Blazor Page Application**

10/05/2020 Developed by Kim Nguyen

10/10/2020 Reviewed by Spencer (Yu Che) Liu

Center for Information Assurance (CIAE) @City University of Seattle (CityU)

 Icon

Description automatically generated 

**Before You Start**

* Version numbers may not match with the most current version at the time of writing. If given the option to choose between stable release (long-term support) or most recent, please choose the stable release rather than beta-testing version.
* This tutorial targets Windows users and MacOS users.
* There might be subtle discrepancies along the steps. Please use your best judgement while going through this cookbook style tutorial to complete each step.
* For your working directory, use your course number. This tutorial may use a different course number as an example.
* The directory path shown in screenshots may be different from yours.
* If you are not sure what to do or confused with any steps:
  1. Consult the resources listed below.
  2. If you cannot solve the problem after a few tries, ask a TA for help.

**Learning Outcomes**

Students will be able to:

* Further understanding of ASP .NET Core Web App – Razor Pages project structures.
* Create a Blazor application using C# ASP .NET
* Run the Blazor application
* Compare, distinguish Razor Pages vs Blazor App.
* Understand Single Page Application

**Resources:**

* Microsoft | ASP .NET - <https://dotnet.microsoft.com/apps/aspnet>
* Blazor University | What is Blazor - <https://blazor-university.com/overview/what-is-blazor/>

**Razor Pages Web App – Structure breakdown**

Last week, we have created a simple web application using dotnet framework. Let’s dive deeper into the project structure:

Text

Description automatically generated

Going from bottom to the top:

* Startup.cs file: contains the app’s services, for example HTTP, and how those services are configured.
* Program.cs file: is the entry point of the program. When we run the application, this file gets to be looked at first by the compiler.
* wwwroot folder: contains static files, such as html, css, javascript. More about static files: <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/static-files?view=aspnetcore-3.1>
* Properties folder: contains launchSettings.json file: decides how the application can be launched, for example, on which port and url the application is available. More about launchSettings.json: <https://docs.microsoft.com/en-us/visualstudio/mac/launch-settings?view=vsmac-2019>

Text

Description automatically generated

* Pages folder: contains html files for each page on the website, where we can customize the front end of the website, what to show when user access the pages. The file extension for html becomes cshtml, which can be understood as html for C# ASP .NET framework.

Graphical user interface, text, application

Description automatically generated

The above structure is known as Razor Pages Application. Razor Pages is a newer, simplified web application programming model. Razor Pages is designed to make common patterns used with web browsers easy to implement when building an app. More about Razor Pages: <https://www.jetbrains.com/dotnet/guide/tutorials/basics/razor-pages/>

**Blazor Web Application:**

1. Now let’s try something new. Let’s create a Blazor Web Application:

Blazor is a Single Page Application development framework. The name Blazor is a combination/mutation of the words Browser and Razor (the .NET HTML view generating engine). The implication being that instead of having to execute Razor views on the server in order to present HTML to the browser, Blazor is capable of executing these views on the client

More about Blazor App: <https://blazor-university.com/overview/what-is-blazor/>

1. In VSCode, open the repository you cloned for this week’s assignment (if you have not, please do before proceeding, instructions are provided in the README.md file in the GitHub repository).
2. Make sure you are in the right path, for example:

/Desktop/CS504/HOP02-YourGitHubUsername/Module 2

1. Open the terminal from the VSCode by hitting the control + ~ key. Type the following command to create a Blazor App:

dotnet new blazorserver -o BlazorApp --no-https

1. You should see this message:

[NOTE: the directory path in the screenshots might look different from yours]

Text

Description automatically generated

And a directory called BlazorApp created for you:

Graphical user interface, text

Description automatically generated

1. Navigate inside the BlazorApp folder:

cd BlazorApp/

1. Run the application:

dotnet run

You should see the following messages on the terminal:

Graphical user interface, text, application

Description automatically generated

1. Ctrl + click on the url (http://localhost:5000) to view the website:

Graphical user interface, text, application, email

Description automatically generated

Next week, we will dive deeper in the structure of Blazor App.

1. **Answer the questions below**, then export this file to PDF (make sure you save in the same Module 2 directory as this word doc):

* What is Single Page Application?
* What is the differences between Razor Pages (created in last week’s HOP) and Blazor App (created in this week’s HOP)?
* What is the benefit of using Blazor App over using Razor Pages?

**Push your work to GitHub**

Open the terminal from the VSCode by hitting the control + ~ key, make sure you are in the right path, for example: /Desktop/CS504/HOP02-YourGitHubUsername/Module 2

Type the following command:

git add . (to copy all changes you have made)

git commit -m “Submission for Module 2 - YourGitHubUsername” (To add a message to your submission)

git push origin master (to upload your work to Github)